Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**Пояснительная записка**К курсовому проектированию

По курсу “Логика и основы алгоритмизации

в инженерных задачах”

на тему “Реализация алгоритма нахождения изолированных вершин в графе”

Выполнила:

студентка группы 19вв2

Гуськова Н.Д

Приняли:

Юрова О .В

Митрохин М.А.

Пенза 2020

**Содержание**

[Реферат 4](#_TOC_250009)

[Введение 5](#_TOC_250008)

1.[Постановка задачи 6](#_TOC_250007)

2.[Теоретическая часть задания 7](#_TOC_250006)

3.[Описание алгоритма программы](#_TOC_250005) 8

4.[Описание программы](#_TOC_250004) 9

5.[Тестирование](#_TOC_250003) 12

6.[Ручной расчёт задачи](#_TOC_250002) 14

[Заключение](#_TOC_250001) 15

[Список литературы](#_TOC_250000) 16

Приложение A. Листинг программы 17

# Реферат

Отчет 18 стр, 7 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, ОРГРАФ.

Цель исследования – Реализация алгоритма нахождения изолированных вершин в графе.

В работе рассмотрена реализация алгоритма нахождения изолированных вершин в графе. Установлено, что с помощью данного алгоритма можно найти изолированные вершины в графе.

# Введение

Алгоритм нахождения изолированных вершин в графе основан на просмотре строк в матрице смежности графа.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2019, язык программирования – С++.

Целью данной курсовой работы является разработка программы на языке С++, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм нахождения изолированных вершин в графе.

# Постановка задачи

Требуется разработать программу, которая находит изолированные вершины в графе.

Исходный граф в программе должен задаваться матрицей смежности,

причём при генерации данных должны быть предусмотрены граничные

условия. Программа должна работать так, чтобы пользователь вводил

количество вершин для генерации матрицы смежности. После обработки этих

данных на экран должна выводиться матрица смежности орграфа, вид орграфа

и все компоненты связности орграфа. Необходимо предусмотреть различные

исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

Устройства ввода – клавиатура, мышь.

Задания выполняются в соответствии с вариантом №12.

# Теоретическая часть задания

Граф G (рисунок 1) задается множеством вершин X1, X2, ..., Xn. и множеством ребер, соединяющих между собой определенные вершины. Ребра из множества А ориентированы, что показывается стрелкой, которая указывает достижимость данной вершины, граф с такими ребрами называется ориентированным графом.

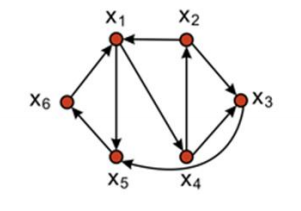


Рисунок 1 - Пример орграфа

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается единицей, иначе нулем.

Изолированная вершина – вершина, которая не имеет инцидентных ребер.

Пример: пусть  – ориентированный граф, , , тогда ,  – изолированные вершины.

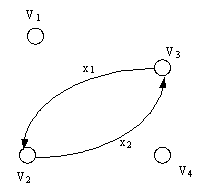


Рисунок 2 – Пример орграфа и изолированными вершиными.

# Описание алгоритма программы

Для программной реализации алгоритма понадобится: двойной указатель int\*\* a – для создания динамической матрицы, которая хранит ориентированный граф, параметр int n, с помощью которого мы задаём размеры графа, int sum для подсчёта количества изолированных вершин.

Создаётся динамическая матрица и случайно заполняется 1 и 0. С помощью функции addisolated(int\*\* a, int n) мы добавляем случайное количество изолированных вершин в наш граф, после выводим его не экран. Далее мы просматриваем каждую вершину в графе, если она изолирована, то выводим ее на экран и увеличиваем значение sum на единицу. После этого, мы выводим количество изолированных вершин.

Ниже представлен псевдокод части программы, которая отвечает за проверку изолированных вершин.

1. для i = 0, пока i < n, i++;
   1. для j = 0, пока j < n, j ++;
      1. если a[i][j] == 1 или a[j][i] == 1, то выходим из цикла;
      2. если j == n – 1, то
         1. sum++;
         2. выводим вершину i на экран

Полный код программы можно увидеть в Приложении А.

# Описание программы

Для написания данной программы использован язык программирования С++. Язык программирования С++ - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

Работа программы начинается с запроса размера графа. После того, как пользователь ввел размеры графа на экран выводится матрица смежности графа. Далее программа добавляет изолированные вершины в граф и выводит его на экран.

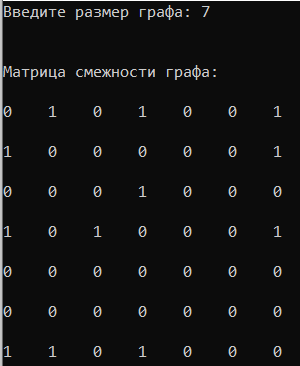


Рисунок 3 Выведенная на экран матрица смежности графа

Далее программа ищет изолированные вершины и выводит их на экран, так же выводится на экран информация о их количестве.

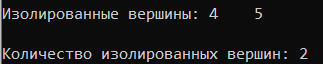


Рисунок 4 информация о изолированных вершинах

Результаты работы программы записываются в файл.

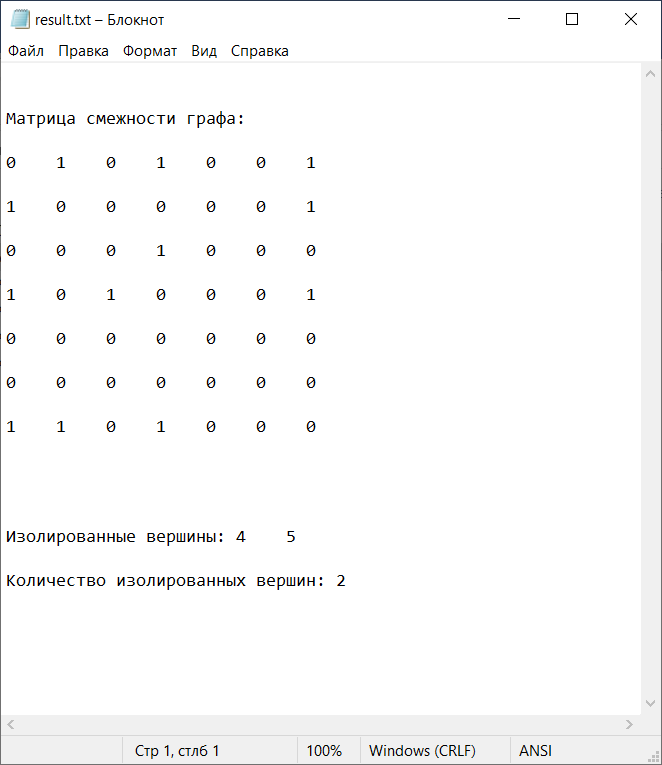


Рисунок 5 файл с результатами работы программы.

# Тестирование

Среда разработки MicrosoftVisualStudio 2019 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

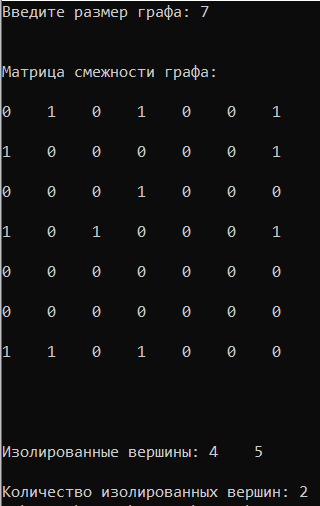


Рисунок 6

На рисунке 6 продемонстрирован результат тестирования при размерах графа 7x7.

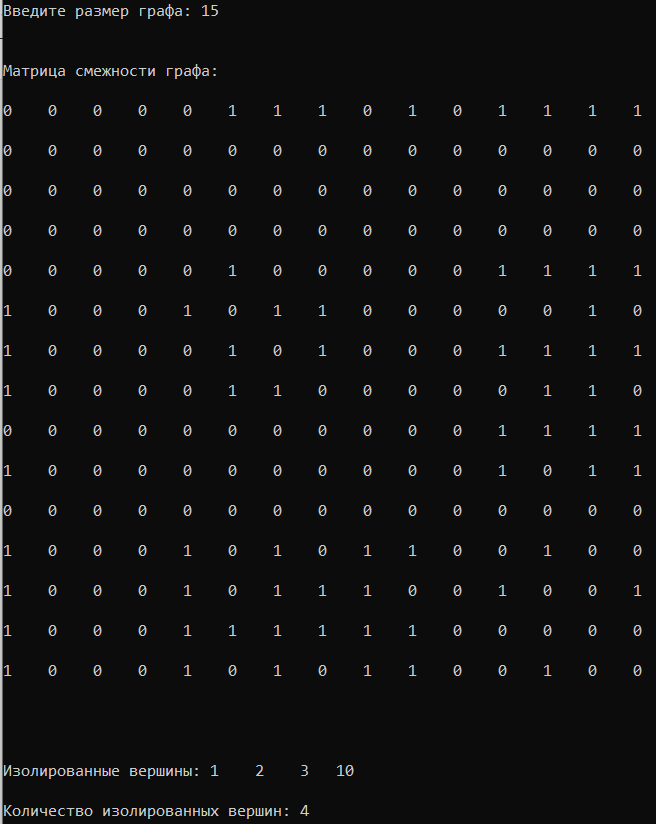


Рисунок 7

На рисунке 7 продемонстрирован результат тестирования при размерах графа 15x15.

В результате тестирования было выявлено, что программа успешно создаёт граф и находит изолированные вершины.

**Ручной расчёт задачи**

Проведем проверку программы посредством ручных вычислений на примере графа с размерами 7x7 (рисунок 6).

Начинаем проверку с 0 строки и 0 столбца матрицы смежности, если находим 1 то прекращаем проверку и переходим к следующему столбцу и строке. В 4, 5 столбце и строке при проверке мы не нашли 1, значит это и есть изолированные вершины.

Сравнив результат работы программ и ручной расчёт, можно сделать вывод, что программа работает верно.

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм поиска изолированных вершин в графе в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежности, основанных на теории орграфов. Приобретены навыки по осуществлению алгоритма поиска изолированных вершин в графа. Углублены знания языка программирования C++.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

* 1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ - М.: МЦНМО, 2001. - 960 с.
  2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978
  3. Герберт Шилдт «Полный справочник по C++» - Вильямс, 2006
  4. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977. 208 с.
  5. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. 2009 г.
  6. 3. Оре О. Графы и их применение: Пер. с англ. 1965. 176 с.

## Приложение А.

**Листинг программы.**

#include <iostream>

#include <iomanip>

#include <fstream>

using namespace std;

void addisolated(int\*\* a, int n)

{

int mod, del = n;

while (del)

{

mod = rand() % n;

for (int i = 0; i < n; i++)

a[i][mod] = a[mod][i] = 0;

del /= 2;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

ofstream fout("result.txt");

int\*\* a, n, sum = 0;

cout << "Введите размер графа: ";

cin >> n;

a = (int\*\*)malloc(n \* sizeof(int\*));

for (int i = 0; i < n; i++)

a[i] = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

if (i == j)

a[j][i] = a[i][j] = 0;

else

a[j][i] = a[i][j] = rand() % 2;

addisolated(a, n); //добавляем изолированные вершины в граф

cout << "\n\nМатрица смежности графа:";

fout << "\n\nМатрица смежности графа:";

for (int i = 0; i < n; i++)

{

cout << "\n\n";

fout << "\n\n";

for (int j = 0; j < n; j++)

{

cout << a[i][j] << setw(5);

fout << a[i][j] << setw(5);

}

}

cout << "\n\n\n\n\nИзолированные вершины: ";

fout << "\n\n\n\n\nИзолированные вершины: ";

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

{

if (a[i][j] == 1 || a[j][i] == 1)

break;

if (j == n - 1)

{

sum++;

cout << i << setw(5);

fout << i << setw(5);

}

}

cout << "\n\nКоличество изолированных вершин: " << sum;

fout << "\n\nКоличество изолированных вершин: " << sum;

fout.close();

}